# AN ADJACENT EXTREME EFFICIENT POINT PROCEDURE FOR BI-CRITERION LINEAR PROGRAMS

By

V. SUNDARESAN



INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAM

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JULY, 1980

# AN ADJACENT EXTREME EFFICIENT POINT PROCEDURE FOR BI-CRITERION LINEAR PROGRAMS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
V. SUNDARESAN

to the

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAM
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JULY, 1980

IMEP-1980-M-SUN-ADJ

# CERTIFICATE

This is to certify that the present work on 'An Adjacent Extreme Efficient Point Procedure for Bi-Criterion Linear Programs', by V. Sundaresan has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

July, 1980

(J.L. Batra)
Professor and Head
Industrial and Management Engg.
Indian Institute of Technology
Kanpur 208016

## ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to Dr. J.L. Batra for the constant guidance and constructive criticisms, rendered throughout the course of the work. His lively interest and readiness to devote his valuable time on my work has been a constant source of inspiration and has also made this work attain its present stature.

I also wish to express my indebtedness to Dr. S. Sadagopan, who had been of immense help to me in carrying out this work with his valuable suggestions.

Words fail to express my gratitude to my friends for the amusing company they gave which made my stay at IIT Kanpur campus a pleasant and memorable one.

I am also thankful to Mr. J.K. Misra for his excellent typing and Mr. Buddhi Ram Kandiyal for his immaculate cyclostyling work.

V. Sundaresan

CONTENTS

| Chapter | | Page |
|---|---|---|

# SYNOPSIS

Decision Making is the process of selecting an alternative among the available ones. When there are more than one criteria for the selection, the problem is called multi-criteria problem. Most real life problems fall under this category. There are various methods to solve this class of problems. All these methods can be classified into four major catagories depending on the preference information provided by the decision maker and the stage at which it is provided.

In this thesis we develop an algorithm for bicriterion linear programs. The algorithm requires posterior articulation of information. The algorithm generates all efficient extreme points in the objective space. Almost all the existing algorithms for these class of problems generate efficient points in the decision space. However, the algorithm proposed by Aneja and Nair [1] concentrates on the generation of efficient points in the objective space. Hence the proposed algorithm is compared with algorithm proposed by Aneja and Nair. For both the algorithms computer codes in FORTRAN IV were developed and implemented on DEC 1090, Computer system.

For computational simplicity, balanced transportation problems of various sizes were developed. These were solved as linear programs. The size of the linear program varied from problems with 7 constraints 16 variable to 13 constraints 49 variables.

The execution times for the generation of one extreme efficient point for both the algorithms were compared for 24 randomly generated problems of 4 different sizes. It was observed that the proposed algorithms is computationally highly efficient as compared to Aneja and Nair's algorithm. For the smallest size problem, i.e., 4 x 4 balanced transportation problem, the average execution time (average based on 6 problems) per extreme efficient point was 1/6 that of the Aneja and Nair's algorithm. For bigger sized problems, the computational efficiency was even still better.

# CHAPTER I

## INTRODUCTION

Decision making is a process of selecting a possible course of action among the various available ones. There may be one or more criteria which will help justify the selection. If there are more than one criterion then the process is called Multiple Criteria Decision Making (MCDM). Normally the criteria are non-commensurable and conflicting. Of late there has been considerable research activity in the area of multi-criteria decision making since most of the real life problems fall under this category. In the MCDM problems, the best alternative is to be selected keeping in the consideration, the various constraints and criteria. In the process of choosing the best alternative, the decision maker is required to give some preference information. The stage at which it is given is the basis on which the MCDM problems can be classified. Th classification of this problems is given in Table 1.1.

The methods for which no articulation of preference information is given, assume that the decision maker will be able to accept the given solution. This assumption is not well justified. The methods where the decision maker is required to give his preference information apriori, suffer from the disadvantage that the decision maker gives this in an information vacuum. The methods which are based

Table 1.1: Classification of Methods for MCDM.

Multiple objective decision making

| Stage at which information is needed | Type of information | Major classes of methods |
|---|---|---|
| No articulation of preference information. | | Global criterion Method |
| A priori articulation of preference information. | Cardinal information | 1. Utility function<br>2. Bounded objective method |
| | Ordinal and cardinal information | 1. Lexicographic method<br>2. Goal programming<br>3. Goal attainment method |
| Progressive articulation of preference information (interactive methods) | Explicit Trade off | 1. Method of Geoffrion and interactive goal programming<br>2. Surrogate worth trade off method<br>3. Method of satisfactory goals<br>4. Method of Zionts-Walleinus |
| | Implicit Trade off | 1. STEM and related methods<br>2. SEMOPS and SIGMOP Methods<br>3. Method of Displaced Ideal<br>4. GPSTEM Method<br>5. Method of Steuer |
| A posteriori articulation of preference information (non-dominated solution generation method). | Implicit Trade off | 1. Parametric method<br>2. -constraint method<br>3. MOLP Methods<br>4. Adaptive Search Method |

on ordinal preference information given apriori depend heavily on this and the moment the decision maker finds that the information given at an earlier time does not hold any more, the problem is to be solved afresh using the new information.

In the third method, which is classified as interactive method, the decision maker is part of the solution process. There is no need of a priori preference information articulation. The major advantage of this is that decision maker learns about the problem as he proceeds. Since decision maker is part of the solution process, the prospects of the obtained solution being implemented, is bright. But the solution depends upon the accuracy of the local preference the decision maker can indicate. This is one of the disadvantages, these methods suffer from. Another disadvantage is that much more effort is required on the part of the decision maker than other methods.

The last category of methods, listed in Table 1.1 are the methods for which a posteriori artifulation of preference information is given. These methods have an advantage over other methods, because no assumption regarding the decision maker's utility function is required.

The present thesis deals with this class of multi-criteria problems. Solving the multi-criteria problem requires the generation of all non-dominated (efficient) points.

The first generalization of the multi-criteria problem is the bicriterion problems. If all the constraints, and both the criteria are linear, then the problem is called bicriteria linear program. The importance of the bicriterion problems can knot be over emphasized. Many situations could be formulated as bicriteria linear programs. For instance in a production system one would like to minimize the cost of production and maximize the production.

A survey of the literature indicates that most of the investigators [2,4,9,10] have approached the bicriteria problem through developing algorithms for generating all the efficient points in the decision space. These algorithms involve the consideration of n dimensional space, where n is the number of decision variables. However, the dimensionality of the solution space can be considerably reduced if instead of the decision space, the objective space is considered. In the present thesis, an algorithm which generates the extreme efficient points in the objective space is developed. The performance of this algorithm is compared with Aneja and Nair's algorithm [1] the only other algorithm reported in the literature which exploits objective space rather than decision space for the generation of extreme efficient points. A set of 24 randomly generated balanced transportation problems of

different sizes are considered for comparing the two algorithms. For both the algorithms, these transportation problems are structured as linear programmes.

The thesis is organized in four chapters. Chapter II deals with problem statement, its formulation and the relevant literature survey. The proposed algorithm is presented in Chapter III. In Chapter IV, a comparison of the proposed algorithm with Aneja and Nair's algorithm is presented. Listing of all the computer programmes developed for this work are given in Appendices A, B and C.

# CHAPTER II

## PPOBLEM STATEMENT AND LITERATURE SURVEY

### 2.1 Problem Statement:

Bicriterion Linear Programs (BLP) may be stated as follows:

Consider a Vector Minimum Problem (VMP)

$$\left\{ Cx, \ Dx \right\}$$
$$\text{s.t. } Ax \leq b$$
$$x \geqslant 0 \qquad (1)$$

where x, C and D are n-dimensional vectors, A is a (m x n) matrix and b is m-dimensional vector. C and D are coefficient of the two criterion functions. A represents the technological coefficients of the activities and b is the requirement vector. $Ax \leq b$ represents the constraints of the linear programme (LP). This statement is made without loss of generality, as maximization problems and greater than or equal to constraints can be readily incorporated using standard techniques of LP.

Let us denote the set of feasible solutions by S and its mapping by the two objectives into objective or criterion space by Y. Y is called the pay off set and is defined as

$$Y = \left\{ (z_1, z_2) \;\middle|\; z_1 = Cx, \ z_2 = Dx \text{ for some } x \in S \right\}$$

A solution is called superior if it minimizes both the objectives simultaneously. In most real life situations superior solutions do not exist. The objectives are non-commensurable. Under these assumptions the decision maker is interested in efficient solutions or non-dominated solutions. For clarity we define efficient solution before proceeding further.

Definition:

A feasible solution $x^o$ is said to be efficient (non-dominated) if $Cx \leq Cx^o$ for some other feasible x implies that $Dx^o < Dx$ and vice versa.

Essentially it means that no other feasible solution is better with respect to both the criterion. Also by implication $x \in S$ is dominated if there exists atleast one feasible solution $x^o$ such that $Cx^o \leq Cx$ and $Dx^o < Dx$. Thus, it is always wise to restrict ourselves to non-dominated solution. The solution to (1) can be interpreted as determination of the set of entire efficient points.

## 2.2 Survey of Literature:

Most of the research in this area has been concentrated on getting all the efficient points in the decision space. One of the early papers in this area is due to Geoffrion [4]. He proposed a scalar maximization approach for obtaining all the efficient solutions. He converted the original problem into the following problem,

Problem $P_\alpha$:

$$\begin{aligned} &\text{Min.} \ \alpha Cx + (1-\alpha)\ Dx \\ &\text{S.t.} \ \ Ax \leqslant b \\ &\qquad\quad x \geqslant 0 \end{aligned} \tag{2}$$

where $\alpha \in (0, 1)$. The problem $P_\alpha$ gives all efficient solutions if $\alpha$ is varied in the given range. He proves the necessary and sufficient conditions in the form of a theorem. Further, Geoffrion suggests that his approach can be extended to multi objective programs with more than two criteria. Bacopoulas and Singer [2] adopted a constraint criteria approach to get all the efficient solution in the decision space. One criteria is converted into a constraint at certain level. By parametrically varying this level he proves that all efficient solution can be obtained.

Steuer [9] has given an algorithm which is a modified version of the simplex algorithm. In the column selection rule stage, the efficiency of the next solution is tested by introducing each possible candidate columns one by one. One LP is solved to test the efficiency or non domination of each point. They call this procedure as adjacent efficient basis algorithm.

Yu and Zeleny [10] have given another modification to the simplex procedure. They refer their modified simplex procedure as adjacent non-dominated basis approach. A

modified version of the column selection rule is used to examine non-domination. Essentially they use the $\bar{c}_j$ of each objective to test non-dominance.

Both Steuer and Zeleny and Yu concentrate on generation of efficient points in the decision space, which requires more computing and book keeping. Further, they have addressed to multi criteria problem in general.

Sadagopan [8] has suggested an interactive solution procedure for the bicriteria mathematical programs. Recently, Aneja and Nair [1] have developed an algorithm basically for bicriterion transportation problems. They claim that their algorithm is applicable to bicriterion linear programs also.

Their algorithm generates efficient extreme points in the objective space. The algorithm requires solving of (2K-3) linear programs if there are K ($\geq 2$) efficient extreme points in the objective space. As the present work considers the same problem and provides algorithm as compared to Aneja and Nair's approach, the details of their algorithm are presented in Chapter III.

The proposed solution methodology involves the generation of efficient extreme points in the criteria space. instead of decision space. This was done for the following reasons. Firstly, the dimensions of the solution space get

reduced if the objective space is considered. For instance in the case of bicriterion problems the objective space is two dimensional while the decision space is n-dimensional where n is the number of decision variables. Secondly, all extreme points in the criteria space correspond to one extreme point in the decision space. Moreover, there may be more than one extreme point in the decision space mapping onto the same extreme point in the objective space. Since the efficient frontier in the objective space is piecewise linear curve, it is enough to restrict ourselves to extreme efficient points. Thus, it is advantageous to look for efficient extreme points in the objective space.

# CHAPTER III

# GENERATION OF EFFICIENT POINTS

## 3.1 Introduction:

In the multicriteria problems as already mentioned, one is interested in the generation of efficient points. It was also stated that efficient points in the objective space will be more useful. Because of the polyhedral property of the payoff set, the set of all efficient solution is determined the moment the extreme points of the polyhedron are known. Since the primary motivation to this research is the algorithm proposed by Aneja and Nair[1], this will be discussed first, followed by the algorithm proposed by us.

## 3.2 Aneja and Nair's Algorithm:

### 3.2.1 Introduction:

Aneja and Nair proposed the following algorithm for generating the entire set of efficient points in the criteria space. Though the authors have concentrated on the determination of efficient extreme points for the bi-criteria transportation problem, they claim that their algorithm is equally applicable to bi-criteria LP problems. The algorithm initially tests for the existence of a superior solution.

If such a solution does not exist, the linear programming problem is solved using objective functions which are positively weighted average of the objective functions. This process is carried out till all the efficient extreme points are determined.

3.2.2 Algorithm:

Step 0: Find $z_1^{(1)} = \text{Min}\ (z_1 \mid x \in S)$ and $z_2^{(1)} = \text{Min}\ (z_2 \mid z_1 = z_1^{(1)}$ and $x \in S)$. Record $(z_1^{(1)}, z_2^{(1)})$ and set $k = 1$. Similarly, find $z_2^{(2)} = \text{Min}\ (z_2 \mid x \in S)$ and $z_1^{(2)} = \text{Min}\ (z_1 \mid z_2 = z_2^{(2)}$ and $x \in S)$. If $(z_1^{(1)}, z_2^{(1)}) = (z_1^{(2)}, z_2^{(2)})$, stop. Else record $(z_1^{(2)}, z_2^{(2)})$ and set $k = k+1$. Define sets $L = \{(1,2)\}$ and $E = \emptyset$, and go to Step 1.

Step 1: Choose an element $(r, s) \in L$ and set

$$a_1^{(r,\ s)} = \left| z_2^{(s)} - z_2^{(r)} \right|$$

$$a_2^{(r,\ s)} = \left| z_1^{(s)} - z_1^{(r)} \right|$$

Let $\bar{x}$ be an optimal solution to the linear program.

$$\text{Min.}\ a_1^{(r,\ s)}\ Cx + a_2^{(r,\ s)}\ Dx$$

$$\text{S.t}\ Ax \leq b$$

$$x \geqslant 0 \qquad (1)$$

If there are alternate optimal choose an optimal solution $\bar{x}$ for which Cx is minimum.

Let $\bar{z}_1 = Cx$ and $z_2 = Dx$

If $(\bar{z}_1, \bar{z}_2)$ is equal either to $(z_1^{(r)}, z_2^{(r)})$ or $(z_1^{(s)}, z_2^{(s)})$ set $E = E \cup \{(r, s)\}$ and go to Step 2. Otherwise record $(z_1^{(k)}, z_2^{(k)})$ such that $z_1^{(k)} = \bar{z}_1$ and $z_2^{(k)} = \bar{z}_2$, and set $k = k+1$.

$$L = L \cup \{(r, k), (k, s)\}$$ and goto Step 2.

<u>Step 2</u>: Set $L = L - \{(r, s)\}$. If $L = \emptyset$ stop. Otherwise goto Step 1.

Aneja and Nair have proved the following results.

<u>Theorem 3.1</u>: A point $z^{(k)} = (z_1^{(k)}, z_2^{(k)})$ is an efficient extreme point in the objective space, if and only if $z^{(k)}$ is recorded by the algorithm.

<u>Theorem 3.2</u>: The algorithm is finite and requires exactly $(2k-3)$ iterations when it has $k(\geq 2)$ efficient extreme points.

3.2.3 <u>Numerical Example</u>:

For better understanding of the Aneja and Nair's algorithm, a numerical example considered by them is discussed. Table 3.1 contains the data for a balanced transportation problem with two cost criteria. The two numbers in each cell represent the cost coefficients. The number in the north west corner indicates $c_{ij}$'s and the numbers in south-east corner denotes $d_{ij}$'s.

Table 3.1: Data for the problem

| | | | | Supply |
|---|---|---|---|---|
| 1<br>4 | 2<br>4 | 7<br>3 | 7<br>4 | 8 |
| 1<br>5 | 9<br>8 | 3<br>9 | 4<br>10 | 19 |
| 8<br>6 | 9<br>2 | 4<br>5 | 6<br>1 | 17 |
| Demand 11 | 3 | 14 | 16 | |

<u>Step 0</u>: $z_1^{(1)} = 143$ $(= z_1 \mid x \in S)$. The solution to the problem of minimizing Cx absolutely over S is 143.

$$z_2^{(1)} = 265$$

The solution is indicated in Table 3.2. The values given by the side denotes the objective function values.

Table 3.2: Optimal solution with $c_{ij}$ cost coefficients

| | | | |
|---|---|---|---|
| 5 | 3 | | |
| 6 | | | 13 |
| | | 14 | 3 |

$z_1^{(1)} = 143 = Cx$

$z_2^{(1)} = 265 = Dx$

Similarly $z_2^{(2)} = 167$ $(= z_2 \mid x \in S)$. The corresponding solution is indicated in Table 3.3. The objective function values are indicated by the side.

Table 3.3: Optimal solution with $d_{ij}$ costs coefficients.

| | | | |
|---|---|---|---|
| | | 8 | |
| 11 | 2 | 6 | |
| | 1 | | 16 |

$z_1^{(2)} = 208$

$z_2^{(2)} = 167$

Iteration No. 1:

Step 1: $(r, s) \in L = (1, 2)$

$$a_1^{(r,\ s)} = 98 \ (= |265 - 167|)$$

$$a_2^{(r,\ s)} = 61 \ (= |208 - 143|)$$

$$\text{Min.} \quad a_1^{(r,\ s)}\, Cx + a_2^{(r,\ s)}\, Dx$$

$$\text{S.t} \quad x \in s \tag{2}$$

The Table 3.4 indicates the modified cost coefficients $a_1^{(r,\ s)}\, c_j + a_2^{(r,\ s)}\, d_j$ for all $j = 1, \ldots mn$ where m and n are the size of the transportation problem.

Table 3.4: Cost for transportation problem in Iteration No. 1.

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Supply |
| | 238 | 456 | 881 | 946 | 8 |
| | 423 | 1402 | 879 | 1042 | 19 |
| | 1174 | 1012 | 717 | 653 | 17 |
| Demand | 11 | 3 | 14 | 16 | |

Table 3.5: Solution for problem corresponding to Table 3.4.

| | | | |
|---|---|---|---|
| 5 | 3 | | |
| 6 | | 13 | |
| | | 1 | 16 |

$z_1^{(3)} = 156$

$z_2^{(3)} = 200$

Solution to the problem (1) is shown in Table 3.5.

Let, $\bar{z}_1 = C\bar{x} = 156$, $\bar{z}_2 = D\bar{x} = 200$

Since $(\bar{z}_1, \bar{z}_2)$ is not either equal to $(z_1^{(1)}, z_2^{(1)})$ or $(z_1^{(2)}, (z_2^{(2)})$, we have, $(z_1^{(3)}, z_2^{(3)}) = (\bar{z}_1, \bar{z}_2)$.

Now, $k = 3+1 = 4$ and $L = \{(1,2), (1,3), (3,2)\}$.

Step 2: $L = L - \{(1,2)\} = \{(1,3), (3,2)\}$.

Iteration No. 2:

Step 1: $(r, s) = (1, 3)$

Summary of further calculations are shown in the form of Table 3.6.

Table 3.6: Bicriteria Solution in the Objective Space.

| Iteration | L | E | Recorded points |
|---|---|---|---|
| 1 | $\{(1,2)\}$ | $\emptyset$ | $z^{(1)} = (143, 265)$ <br> $z^{(2)} = (208, 167)$ |
| 2 | $\{(1,3),(3,2)\}$ | $\emptyset$ | $z^{(3)} = (156, 200)$ |
| 3 | $\{(3,2)\}$ | $\{(1,3)\}$ | - |
| 4 | $\{(3,4), (4,2)\}$ | $\{(1,3)\}$ | $z^{(4)} = (176, 175)$ |
| 5 | $\{(4,2)\}$ | $\{(1,3), (3,4)\}$ | - |
| 6 | $\{(4,5), (5,2)\}$ | $\{(1,3), (3,4)\}$ | $z^{(5)} = (186, 171)$ |
| 7 | $\{(5,2)\}$ | $\{(1,3),(3,4),(4,5)\}$ | - |
| 8 | $\emptyset$ | $\{(1,3),(3,4),(4,5),(5,2)\}$ | - |

A computer program was developed to solve the problems by Aneja and Nair's method. The solution shown in Table 3.6 required seven linear program to be solved.

In the following section, the proposed algorithm for generating the efficient extreme points in the objective space is described.

## 3.3 An Adjacent Extreme Point Procedure:

### 3.3.1 Background:

In Chapter II, it was pointed out that the determination of efficient extreme points in objective space

rather than the decision space offers two important advantages in case of multicriteria problems. Algorithm described in section 3.2 generates all efficient extreme points in the objective space. Even though the convexity of the efficient frontier is recognized by Aneja and Nair [1], it is not fully exploited. It was also mentioned that 2k-3 linear programs need to be solved if k ($k \geq 2$) efficient extreme points exists.

In this section, an algorithm is developed which attempts to secure an adjacent efficient extreme point in each pivot operation. As explained later, the algorithm may not always be successful in generating the adjacent efficient extreme points. However, this does not destruct the utility of the algorithm since the next extreme efficient point will be secured in few more pivot steps. Before we describe the actual algorithm, certain preliminary results which lay the foundation for the development of the algorithm are stated.

### 3.3.2 Preliminary Results:

Let us define the original problem as

$$\begin{array}{lll} PO: & \text{Minimize} & \{CX,\ DX\} \\ & \text{s.t.} & AX \leq b \\ & & X \geq 0 \end{array}$$

where C, and D are n-dimensional vectors representing the cost coefficients, of the two objectives. X is a

n-dimensional vector of variables. A is m x n matrix representing technological coefficients and b is m x 1 vector of requirements.

Consider the following single objective, mathematical programs

$$P1 : \text{Minimise } CX \quad \text{s.t. } AX \leq b,\ X \geq 0 \qquad P2 : \text{Minimise } DX \quad \text{s.t. } AX \leq b,\ X \geq 0$$

Let the optimal values of P1 and P2 be $z_1$ and $z_2$ respectively. Problems $P_{z_1}$ and $P_{z_2}$ are defined as follows:

$$P_{z_1} : \text{Minimise } DX \quad \text{s.t. } AX \leq b,\ CX \leq z_1,\ X \geq 0 \qquad P_{z_2} : \text{Minimise } CX \quad \text{s.t. } AX \leq b,\ DX \leq z_2,\ X \geq 0$$

Let $x_1^*$ and $x_2^*$ be the optimal solutions for the problems $P_{z_1^*}$ and $P_{z_2^*}$ respectively. Sadagopan [8] has suggested a procedure for reducing the PO problem to $P_{z_1}$ ($P_{z_2}$) problem with parametrization of $z_1(z_2)$. The procedure utilizes the following theorems.

Theorem 3.3:

In the optimal solution to the program $P_{\bar{z}_2}$ where $\bar{z}_2 \in [z_2^*, Dx_1^*]$ the constraint $DX \leq \bar{z}_2$ will be a binding one i.e. if $\bar{x}$ solve $P_{\bar{z}_2}$ then $D\bar{x} = \bar{z}_2$.

Proof:

Assume the contrary i.e.

$$D\bar{x} < \bar{z}_2 \leq Dx_1^* \tag{3}$$

$z_1^*$ being the absolute minimum of C over S

$$z_1^* = Cx_1^* \leq C\bar{x} \tag{4}$$

Two cases will be considered.

Case 1: Let $Cx_1^* = C\bar{x}$ (5)

(3) and (5) taken together contradict the fact that $x_1^*$ solves $P_{z_1^*}$ since $\bar{x}$ is a better solution.

Case 2: Let $Cx_1^* < C\bar{x}$ (6)

Consider the line segment $[\bar{x}, x_1^*]$ which lies in the convex set S. Over this line segment the linear functions C and D are non-increasing. Because of strict inequality in (6), there exists a feasible solution x' on the line segment $[\bar{x}, x_1^*]$ such that

$$\begin{aligned} D\bar{x} &< Dx' < \bar{z}_2 \\ C\bar{x} &> Cx' > Cx_1^* \end{aligned} \tag{7}$$

Inequality (7) indicates that x' is a feasible solution to $P_{\bar{z}_2}$ with $Cx' < C\bar{x}$ which contradicts the optimality of $\bar{x}$.

Hence $D\bar{x} = \bar{z}_2$.

Theorem 3.4:

$\bar{x} \in S$ is efficient if and only if $\bar{x}$ solves $P_{\bar{z}_2}$ where $\bar{z}_2 \in [z_2^*, Dx_1^*]$.

The sufficiency and necessity of this theorem is discussed below.

Sufficiency: Let $\bar{x}$ solve $P_{\bar{z}_2}$ (henceforth whenever not mentioned $\bar{z}_2 \in [z_2^*, Dx_1^*]$). If $\bar{x}$ is to be efficient, then it needs to be shown that there does not exist any $x \in S$ such that,

$$\begin{aligned} &\text{Case (i)} \quad Dx < D\bar{x} \quad \text{and } Cx \leq C\bar{x} \\ &\qquad\qquad\qquad \text{or} \\ &\text{Case (ii)} \quad Cx < C\bar{x} \quad \text{and } Dx \leq D\bar{x} \end{aligned} \tag{8}$$

Case (i): Let there be an $x$ ($x \in S$) such that $Dx < D\bar{x} = \bar{z}_2$. Solution $x$ is a feasible solution to $P_{\bar{z}_2}$ but not optimal since the theorem 3.3 states that every optimal solution $x^o$ satisfies $Dx^o = \bar{z}_2$. Hence $Cx > C\bar{x}$. Therefore, $x$ is not efficient.

Case (ii): Consider some $x \in S$ such that $Cx < C\bar{x}$, where $\bar{x}$ solves $P_{\bar{z}_2}$. Since $\bar{x}$ is optimal [illegible] $x$ cannot be feasible. (otherwise $x$ would have solved $P_{\bar{z}_2}$). This results in $Dx > D\bar{x}$. Consequently $x$ is not efficient.

Thus $\bar{x}$ is efficient.

<u>Necessity</u>: Let E be the set of all efficient solutions. Suppose $\bar{x} \in E$. Obviously $C\bar{x} \leq Cx_2^*$ as otherwise $x_2^*$ will dominate $\bar{x}$. Let $C\bar{x} = \bar{z}_1$. Assume that $\bar{x}$ does not solve $P_{\bar{z}_1}$ but some other $x' \in S$ solves $P_{\bar{z}_1}$. By theorem 3.3, $C\bar{x} = \bar{z}_1 = Cx'$. Since $x'$ is optimal to $P_{\bar{z}_1}$, $Dx' \lneq D\bar{x}$. However, because $\bar{x}$ belongs to E this implies that $Dx' \geqq D\bar{x}$. Hence $Dx' = D\bar{x}$. Conseqcuently $\bar{x}$ solves $P_{\bar{z}_1}$.

Thus it is possible to generate the entire set of efficient solutions by parametrically solving $P_{\bar{z}_2}$. By theorem 3.3, the generated solution will have specific levels of attainment $(\bar{z}_2)$, of the second criterion. Hadley [5] suggests that for the parametric variation of the right hand side of the constraint set, dual simplex algorithm can be used.

3.3.3 <u>Algorithm (Adjacent Extreme Point Procedure)</u>:

<u>Step 0:</u> Solve P1 and P2. Let $z_1^*$ and $z_2^*$ be the optimal solutions to P1 and P2 respectively. Solve $P_{z_1^*}$ and $P_{z_2^*}$. Let $x_1^*$ and $x_2^*$ represent the corresponding optimal solutions.

<u>Step 1:</u> Let $\bar{C}$ and $\bar{D}$ be the relative cost vectors of the objective function of problem P1 whose cost vectors are C and D. (In the first iteration, if alternate optimum to P1 exists then choose among the alternate optimum, that solution which minimizes the second objective with cost vector D). Determine

an adjacent extreme point solution in which the entering variable in $x_i$ such that

$$x_i = \operatorname*{Min}_{j} \left\{ -\frac{\bar{c}_j}{\bar{d}_j} \text{ for } \bar{c}_j < 0 \text{ and } \bar{d}_j > 0 \right\} \quad (9)$$

(In case of tie break them arbitrarily).

Set $k = k+1$, record $z_1^{(k)} = CX$, $z_2^{(k)} = DX$

If $DX = Dx_2^*$, stop. Otherwise, go to Step 1.

<u>Theorem 3.5</u>:

Algorithm described above generates all efficient extreme points.

<u>Proof</u>: As mentioned in section 3.3.2, the basic idea of this algorithm is to generate all efficient extreme points by solving $P_{\bar{z}_2}$ over $\bar{z}_2 \in [z_2^*, Dx_1^*]$. Only one constraint ($Dx \leq \bar{z}_2$) is added to P1. Hence it can be treated as right hand side parametrization problem for the added constraint. Dual simplex algorithm can be used to do the parametrization.

Let B be the basis matrix associated with the problem P1. The matrix A with objective function in cononical form is represented as follows:

$$\begin{bmatrix} C_B & C_N \\ B & N \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} z \\ b \end{bmatrix} \quad (10)$$

where B (m x m) and N (m x n-m) are basic and not basic

matrices of A. C is also divided into $C_B$ (1 x n) and $C_N$ (1 x n-m) representing the basic and non-basic cost vectors respectively. With the addition of the constraint $Dx \leq \bar{z}_2$, the constraints can be written as follows.

$$\begin{aligned} BX_B + NX_N + 0.s &= b \\ D_B X_B + D_N X_N + 1.s &= \bar{z}_2 \end{aligned} \qquad (11)$$

Let A' and b' be denoted as in (12) and (13) respectively. $B^*$ represents the basis of $P_{\bar{z}_2}$. Then

$$A' = \begin{bmatrix} B & N & 0 \\ D_B & D_N & 1 \end{bmatrix} \qquad (12)$$

$$b' = \begin{bmatrix} b \\ \bar{z}_2 \end{bmatrix} \qquad (13)$$

$$B^* = \begin{bmatrix} B & 0 \\ D_B & 1 \end{bmatrix} \qquad (14)$$

Taking the inverse of $B^*$ [6], we have

$$B^{*-1} = \begin{bmatrix} B^{-1} & 0 \\ -D_B B^{-1} & 1 \end{bmatrix} \qquad (15)$$

Thus $B^{*-1}A'$ is as follows:

$$\begin{bmatrix} B^{-1} & 0 \\ -D_B B^{-1} & 1 \end{bmatrix} \begin{bmatrix} B & N & 0 \\ D_B & D_N & 1 \end{bmatrix} = \begin{bmatrix} I & B^{-1}N & 0 \\ 0 & D_N - D_B B^{-1}N & 1 \end{bmatrix} \qquad (16)$$

where $D_B$ (1xm) and $D_N$(1 x n-m) are basic and non-basic cost vectors of D respectively.

To determine the right hand side range of the added constraint, the system $B^{*-1}$ b' $\geqslant$ 0 is to be solved. It essentially corresponds to the condition

$$\begin{bmatrix} B^{-1} & 0 \\ -D_B B^{-1} & 1 \end{bmatrix} \begin{bmatrix} b \\ \xi^{(k)} \end{bmatrix} \geqslant 0 \qquad (17)$$

where $\xi^{(k)}$ represents the range over which $P_{\bar{z}_2}$ continues to represent the optimal basis. Hence the range will be $[\underline{z}_2, \bar{z}_2]$, where $\bar{z}_2$ is the upper limit and $\underline{z}_2$ is the lower limit of $\xi^{(k)}$ by solving the system (17).

The variable to leave the basis will be the slack variable corresponding to the last row and the variable to enter the basis will be decided by the minimum ratio rule of the dual simplex method. Since $D_N - D_B\, B^{-1}\, N$ precisely corresponds to the relative cost associated with the objective function with cost coefficients of D vector, the rule to decide the incoming variable in Step 1 of the proposed algorithm corresponds to the minimum ratio rule.

Theorem 3.6:

Algorithm described in Section 3.3.3 terminates in finite number of iterations.

Proof: In dual simplex, no basis is ever repeated. In fact the previous basis becomes infeasible to successive ones. The number of extreme points in the objective space is finite. Hence the algorithm terminates in finite number of iterations.

It should be noted that there may exist tie for the entering variable at some iterations, i.e., more than one variable has the minimum ratio of $-\bar{c}_j/\bar{d}_j$ where $\bar{c}_j$ and $\bar{d}_j$ are dual costs of non-basic variables. The tie can be broken arbitrarily. In such cases, next adjacent extreme efficient point may not be obtained in one pivot operation. In case degeneracy is encountered at some iterations, the lexicographic rule of simplex method can be adopted to prevent cycling.

### 3.3.4 Numerical Example:

The balanced transportation problem which was given as an illustrated example in Section 3.2.3, is discussed to bring out clearly, the features of the proposed algorithm. Data for the problem is given in Table 3.1.

Solution:

Step 0: Solving P1 and P2, we have $z_1^* = 143$, $z_2^* = 167$. The corresponding solutions appear in Table 3.2 and Table 3.3.

Iteration No. 1:

Step 1: Starting with the solution given in Table 3.2, the dual costs $\bar{c}_{ij}$ and $\bar{d}_{ij}$ of the non-basic cells are

determined. They are shown in Table 3.7.

Table 3.7: Dual costs for the transportation problems at iteration no. 1.

| | | | | | $u_i^1$ | $u_i^2$ |
|---|---|---|---|---|---|---|
| | (5) | (3) | 5 / -10 | 3 / -5 | 0 | 0 |
| | (6) | 7 / 3 | 1 / -5 | (13) | 0 | 1 |
| | 5 / 10 | 5 / 6 | (14) | (3) | 2 | -8 |
| $v_j^1$ | 1 | 2 | 4 | 4 | | |
| $v_j^2$ | 4 | 4 | 13 | 9 | | |

$z_1^{(1)} = 143$
$z_2^{(1)} = 265$

In Table 3.7, the numbers in the north west corner indicates the dual cost $\bar{c}_{ij}$ and the numbers in the south east corner represents the dual cost $\bar{d}_{ij}$. $u_i^1$ and $v_j^1$ around the table represents $u_i$ and $v_j$ with respect to $c_{ij}$ costs. $u_i^2$ and $v_j^2$ are $u_i$ and $v_j$ with respect to $d_{ij}$ costs.

Cell (2,3) enters the basis in the next iteration since the ratio of $-\bar{c}_{ij}/\bar{d}_{ij}$ is minimum for this non-basic cell.

Now $k = 0+1$, $z_1^{(k)} = 143$, $z_2^{(k)} = 265$.

Since $z_2^{(k)}$ is not equal to 167 ($Dx_2^*$) go to Step 1.

Iteration No.2:

Step 1: Now the dual costs of non-basic variables are revised. They appear in Table 3.8 with the objective function values by the side.

Table 3.8: Dual costs of non-basic cells at Iteration No.2.

| | | | | | $u_i^1$ | $u_i^2$ |
|---|---|---|---|---|---|---|
| | 5 | 3 | 4 -5 | 2 0 | 0 | 0 |
| | 6 | 7 3 | 13 | 2 5 | 0 | 1 |
| | 6 5 | 6 1 | 1 | 16 | 1 | -3 |
| $v_j^1$ | 1 | 2 | 3 | 5 | | |
| $v_j^2$ | 4 | 4 | 8 | 4 | | |

$z_1^{(2)} = 156$

$z_2^{(2)} = 200$

Now $k = k+1 = 2$, $z_1^{(2)} = 156$, $z_2^{(2)} = 200$.

Further calculations are shown in Table 3.9.

Table 3.9: Solution to the transportation problem using the proposed algorithm.

| Iteration No. | Leaving Variable | Entering Variable | Sl.No. of extreme efficient point (k) | $z^{(k)}$ |
|---|---|---|---|---|
| 1 | (2,4) | (2,3) | 1 | (143, 265) |
| 2 | (1,1) | (1,3) | 2 | (156, 200) |
| 3 | (3,3) | (3,2) | 3 | (176, 175) |
| 4 | (1,2) | (2,2) | 4 | (186, 171) |
| 5 | - | - | 5 | (208, 167) |

The set of efficient extreme points were generated in 5 iterations. Only two linear programs were solved and the rest of the calculations were only pivot operations. Whereas Aneja and Nair method requires solving of seven linear programs to arrive at the same set of extreme efficient points.

# CHAPTER IV

## RESULTS AND DISCUSSIONS

Computer codes were developed for the proposed as well as the Aneja and Nair's algorithms. The programmes were written in FORTRAN IV and implemented in DEC 1090 computer system. Since both the algorithms require solving of linear programs, a standard linear program code from International Mathematical and Statistical library was utilized. The listings of programmes for Aneja and Nair's algorithm and the proposed algorithm are given in Appendix A and B respectively. The execution times for the randomly generated problems of various sizes were recorded for both the algorithms along with the number of extreme efficient points generated. It needs to be pointed out that the lexicographic rule required to prevent cycling due to degeneracy was not incorporated in the computer code. This results in repetitive generation of certain points. However, very few points exhibited this characteristics. The tie for entering variable in the proposed algorithm is resolved using the strategy of steeper reduction. This means that of all the non-basic variables, that $x_j$ is chosen to enter the basis for which the product $\bar{c}_j\,\theta$ is maximum. Here $\theta$ represents the level at which $x_j$ enters the basis.

## 4.1 Generation of Random Problems:

In order to generate feasible and bounded linear programs, balanced transportation problems were generated. The cost coefficients generated were in the interval 1 to 9. The demand and supply were in the interval 1 to 99 for smaller problems and 1 to 15 for bigger problems. A separate Computer programme was developed for the generation of random problems. The programme listing is given in Appendix C.

The randomly generated balanced transportation problem of size m x n, is converted into a linear program with m + n - 1 constraints and mn variables. For example, a 4 x 4 balanced transportation problem results in a linear program with 7 constraints and 16 variables.

## 4.2 Computation Time:

Computational experience for 24 randomly generated balanced transportation problems was gathered for both the algorithms. Problems of four different sizes viz., 4 x 4, 5 x 5, 6 x 6 and 7 x 7 were generated. Further, six problems for each problem size were considered. The computational times for solving these problems (in case of both the algorithms) are given in Table 4.1. For each problem size, Table 4.2 gives the average execution times for the generation of one efficient point for both the algorithms. The last column of this table gives a ratio measure to compare the computational efficiency of the two algorithms

Table 4.1: Execution times by the proposed Algorithm for problems examined.

| Problem Size | | No. of efficient extreme points | Time taken for generation of extreme efficient points by Aneja and Nair's algo. (time in $10^{-3}$sec.) | No. of points generated by the proposed algorithm. | Time taken for generation of efficient points (time in $10^{-3}$sec.) |
|---|---|---|---|---|---|
| 4x4 | 1 | 8 | 3397 | 8 | 394 |
| | 2 | 2 | 839 | 2 | 344 |
| | 3 | 2 | 2779 | 2 | 381 |
| | 4 | 5 | 2317 | 6 | 384 |
| | 5 | 4 | 2060 | 4 | 379 |
| | 6 | 3 | 1078 | 3 | 283 |
| 5x5 | 1 | 2 | 1645 | 2 | 665 |
| | 2 | 4 | 4008 | 4 | 770 |
| | 3 | 5 | 5260 | 5 | 810 |
| | 4 | 6 | 7351 | 6 | 925 |
| | 5 | 7 | 8465 | 7 | 869 |
| | 6 | 9 | 9220 | 9 | 808 |
| 6x6 | 1 | 4 | 8418 | 4 | 1642 |
| | 2 | 4 | 8181 | 5 | 1781 |
| | 3 | 8 | 17112 | 8 | 1866 |
| | 4 | 9 | 20723 | 9 | 2129 |
| | 5 | 10 | 22685 | 13 | 2286 |
| | 6 | 13 | 30853 | 14 | 2388 |
| 7x7 | 1 | 6 | 18155 | 6 | 2321 |
| | 2 | 7 | 28897 | 7 | 2337 |
| | 3 | 9 | 30857 | 11 | 2588 |
| | 4 | 9 | 29105 | 9 | 2326 |
| | 5 | 11 | 38964 | 12 | 2686 |
| | 6 | 12 | 42863 | 14 | 2957 |

for the generation of one efficient extreme point. Of the 24 problems considered, for three problems, there was repetitive generation of the same point at some iterations. For one problem of size 6 x 6 there were three repetitions while for two problems of 7 x 7 there were two repetitions each. As mentioned earlier, the lexicographic rule to avoid cycling due to degeneracy was not incorporated in the computer code and this resulted in the repetitive generation of these points. Further whenever a tie existed for the entering variable the next adjacent extreme efficient point was not generated in one pivot operation. In case of tie a point was obtained on the boundary line joining the present extreme point and the next extreme adjacent efficient point. This phenomenon was observed in one problem of size 4 x 4, two problems size 6 x 6 and one problem of problem size 7 x 7. Only one boundary point was generated in each of these problems.

The average execution time for one efficient point is calculated by dividing the total execution time for all the six problems of particular size by the number of efficient extreme points as determined by Aneja and Nair's algorithm. The second and third column of Table 4.2 show the average execution time for one efficient point by Aneja and Nair's algorithm and the proposed algorithm.

Table 4.2: Average execution times for generation of one efficient point by both the algorithms.

| No. of problems | Size | Average time for generation of one efficient point by Aneja and Nair's algorithm. (A) | Average time for generation of one efficient point by proposed algorithm (B) | Ratio (A)/(B) |
|---|---|---|---|---|
| 6 | 4 x 4 | 445.35 | 77.32 | 5.75 |
| 6 | 5 x 5 | 1089.36 | 146.88 | 7.41 |
| 6 | 6 x 6 | 2249.38 | 251.91 | 8.92 |
| 6 | 7 x 7 | 2777.48 | 281.38 | 9.87 |

From Table 4.2, it is clear that the proposed algorithm is computationally superior to Aneja and Nair's algorithm. For the 4 x 4 problem, the later algorithm required approximately 6 times more execution time. As the problem size increases, the proposed algorithm shows even better computational superiority over the Aneja and Nair's algorithm. Aneja and Nair's algorithm requires solving 2k-3 linear programs, where k ($\geq 2$) efficient extreme point exists, whereas the proposed algorithm requires only two linear programs and a little more than k pivot operations. This explains the reason for the computational superiority of the proposed algorithm over the Aneja and Nair's algorithm.

It should be noted that the above stated computer times are based on solving transportation problems using LP code. Further computational time reduction can be envisaged if a good transportation code is utilized, instead of the general linear programming code. This is due to the fact, the transportation algorithm will not require the calculation of the dual cost for all the variables after each pivot operation. Only the $u_i$'s and $v_j$'s along with the corresponding row and column involved in the pivot operation need to be computed again.

## 4.3 Conclusions:

In this thesis, an algorithm which generates efficient points in the objective space is developed to solve bicriterion linear programs.

The algorithm exploits the convexity of the efficient frontier in the objective space. The original bicriterion problem is reduced to a problem of parametrization using the results of Sadagopan [8]. This resulted in the proposed algorithm to require solving of two linear programs and a little over k pivot operations where k is the number of efficient points. On the other hand, Aneja and Nair's algorithm needs the solution of 2k-3 linear programs as the authors do not fully exploit the convexity of the efficient frontier.

The comparison of the average execution time to generate one efficient extreme point by both the algorithms clearly indicated the computational superiority of the proposed algorithm over Aneja and Nair's algorithm. For instance, the smallest problem i.e., 4 x 4 balanced transportation problem required only one sixth of the average execution time of Aneja and Nair's algorithm. The computational superiority is even more significant when larger problems are considered.

For very large problems number of efficient points may be large. In these cases an interactive procedure could be developed, which incorporates the proposed algorithm, so that the search region of the efficient frontier is reduced.

This algorithm as such can not be extended to problems with more than two criteria. However, there is a need to develop algorithms for generation of efficient points in the objective space for multicriteria problems.

## REFERENCES


1. Aneja, Y.P., K.P.K. Nair, 'Bicriteria Transportation Problems', Management Science, 25, 73-78, 1979.

2. Bacopoulas, A and I.Singer, 'On Convex Vectorial Optimization in Linear Spaces', Jl. Opt. Theory and Appl. 21, 175-188, 1977.

3. Ching-Lai Hwang and Abu Syed Md. Masud, 'Multiple Objective Decision Making - Methods and Applications Springer-Verlag, New York, 1979.

4. Geoffrion, A.M., 'Solving Bicriterion Mathematical Programs', Operations Research, 15, 39-54, 1967.

5. Hadley, G.H., Linear Programming , pp. 384, Addison-Wesley, 1978.

6. Hadley, G.H., Linear Programming, pp. 36, Addison-Wesley, 1978.

7. Hadley, G.H., Linear Programming, pp. 422, Addison-Wesley, 1978.

8. Sadagopan, S. and A.Ravindran, 'Interactive Solution of Bicriteria Mathematical Programs' Research Memorandum No. 80-2, School of Industrial Engineering, Purdue University, 1980.

9. Steuer, R.E., 'Linear Multiple Objective Programming: Theory and Computational Experience, Ph.D. Dissertation, University of North Carolina, 1973.

10. Yu, P.L. and M. Zeleny, 'Linear Multiparametric Programming by Multicriteria Simplex Method', Management Science, 23, 159-170, 1976.

## APPENDIX A

## Program Listing for the Proposed Algorithm

```
C
C*** MAIN ROUTINE
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
      COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
      COMMON /B11/BASIS(50),IBASIS(50)
      COMMON /B13/IZCNT,IZ(50,2)
      COMMON /B6/X(50,50)
      COMMON /IO/INPUT,IOUT
      COMMON /IO1/IOUT1,INPUT1
      COMMON /B25/ALPHA(5)
      LOGICAL END
      INTEGER BASIS,DBASIS,TIME1
      DATA INPUT,IOUT,IOUT1,INPUT1/20,5,22,23/
      DATA IA/50/
      END=.FALSE.
      NINE=9
      OPEN (UNIT=INPUT,FILE='TRANSP.DAT')
      OPEN (UNIT=IOUT,FILE='TRANSP.OUT')
      OPEN (UNIT=IOUT1,FILE='TIMING.REP',ACCESS='APPEND')
      OPEN (UNIT=INPUT1,FILE='TIMING.OLD')
C     TYPE 401
C 401 FORMAT( 'Give the number of problems')
C     ACCEPT *,NOPROB
C     DO 1200 IIJJ=1,NOPROB
  1   CALL RTIME(TIME1)
      CALL ZERO
      CALL DREAD(END)
      IF(END) GO TO 1000
      CALL SETUP
      CALL FIRST
      CALL SECOND
C     TYPE *,IZCNT
      CALL TIME(TIME1,IZCNT)
      CALL PRINT(NINE)
      NINE=9
      GO TO 1
 1000 STOP
      CALL UERTST
      CALL ZX1LP
      END
C
C***  FIRST ***
C
      SUBROUTINE FIRST
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,.
      COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
      COMMON /B6/X(50,50)
      COMMON /B13/IZCNT,IZ(50,2)
      DIMENSION PSOL(50),DSOL(50),COST(50)
      INTEGER ROW,COL
      M1=M+N-1;M2=M*N
      DO 10 I = 1,M2
      PSOL(I)=0.0
      DSOL(I)=0.0
      COST(I)=0.0
      ROW=(I-1)/N+1
      COL=I-((ROW-1)*N)
      COST(I)=-ICOST1(ROW,COL)
   10 CONTINUE
      CALL ZX3LP(A,IA,B,COST,M2,0,M1,S,PSOL,DSOL,RW,IW,IER)
      DO 15 I = 1,M
      DO 15 J = 1,N
      IJ=((I-1)*N+J)
      A(M1+1,IJ)=-COST(IJ)
      COST(IJ)=-ICOST2(I,J)
```

```
 15     CONTINUE
        B(M1+1)=-S
        CALL ZX3LP(A,IA,B,COST,M2,0,M1+1,S,PSOL,DSOL,RW,IW,IER
        CALL MATFOM(PSOL)
        CALL XCHANG(PSOL,COST,X,M2,SUM)
C       CALL PRINT(2)
        TEMP1=0.0
        TEMP2=0.0
        DO 20 I = 1,M
        DO 20 J = 1,N
        TEMP1=TEMP1+ICOST1(I,J)*X(I,J)
        TEMP2=TEMP2+ICOST2(I,J)*X(I,J)
 20     CONTINUE
        IZ(1,1)=TEMP1+0.5
        IZ(1,2)=TEMP2+0.5
        IZCNT=1
        CALL PRINT(13)
        RETURN
        END
C
C*** SECOND ***
C
        SUBROUTINE SECOND
        COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
        COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
        COMMON /B11/BASIS(50),IBASIS(50)
        COMMON /B13/IZCNT,IZ(50,2)
        COMMON /B6/X(50,50)
        INTEGER ENTER,LEAVE
        LOGICAL STOP,UBCOND
        STOP=.FALSE.
        M1=M+N-1;M2=M*N
 5      CALL FNDENT(ENTER,STOP)
        IF(STOP)GO TO 100
        CALL FDLEAV(ENTER,LEAVE,UBCOND)
        IF(UBCOND) CALL ERROR(1)
        CALL PIVOT(M1,M2,A,B,BASIS,ENTER,LEAVE)
        CALL COMPUT
        GO TO 5
 100    RETURN
        END
C
        SUBROUTINE PIVOT(M,N,A,B,BASIS,KEYCOL,KEYROW)
        DIMENSION A(50,50),B(50),BASIS(50)
        INTEGER BASIS
        DENOMR=A(KEYROW,KEYCOL)
        DO 3 J = 1,N
        IF(A(KEYROW,J).EQ.1.0)2,3
 2      DO 4 I =1,M
        IF(I.EQ.KEYROW)GO TO 4
        IF(A(I,J).NE.0.0)GO TO 3
 4      CONTINUE
        ICHANG=J
        GO TO 5
 3      CONTINUE
        TYPE 505
 505    FORMAT(' SOME THING WRONG IN 51290')
 5      DO 10 J=1,N
        A(KEYROW,J)=A(KEYROW,J)/DENOMR
        B(KEYROW)=B(KEYROW)/DENOMR
 10     CONTINUE
        DO 30 I = 1,M
        IF(I.EQ.KEYROW)GO TO 30
            DO 20 J=1,N
            IF (J.EQ.KEYCOL)GO TO 20
            A(I,J)=A(I,J)-A(I,KEYCOL)*A(KEYROW,J)
 20         CONTINUE
            B(I)=B(I)-A(I,KEYCOL)*B(KEYROW)
 30     CONTINUE
        DO 35 I =1,M
```

```
      A(I,KEYCOL)=0.0
      IF(I.EQ.KEYROW)A(KEYROW,KEYCOL)=1.0
      IF(BASIS(I).EQ.ICHANG)BASIS(I)=KEYCOL
   35 CONTINUE
      RETURN
      END
C
      SUBROUTINE XCHANG(PSOL,COST,X,N,SUM)
      COMMON /B1/NOSP,NOSHRT,MA(50),MB(50),ICOST1(50,50),ICOS
      DIMENSION PSOL(N),COST(N),X(50,50)
      INTEGER ROW,COL
      SUM=0.0
      DO 30 I = 1,N
      ROW=(I-1)/NOSHRT +1
      COL=I-((ROW-1)*NOSHRT)
      X(ROW,COL)=PSOL(I)
      SUM=SUM + PSOL(I)*COST(I)
   30 CONTINUE
      RETURN
      END
      SUBROUTINE ZERO
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
      COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
      COMMON /B11/BASIS(50),IBASIS(50)
      COMMON /B13/IZCNT,IZ(50,2)
      COMMON /B6/X(50,50)
      DO 10 I = 1,50
      MA(I)=0;MB(I)=0
      B(I)=0
      BASIS(I)=0
      IBASIS(I)=0
      IZ(I,1)=0;IZ(I,2)=0
      DO 10 J = 1,50
      A(I,J)=0.0
      X(I,J)=0.0
   10 CONTINUE
      M=0;N=0;IZCNT=0
      DO 15 I = 1,4000
      RW(I)=0.0
   15 CONTINUE
      DO 20 J = 1,2000
   20 IW(J)=0
      RETURN
      END
      SUBROUTINE TIME(TIME1,IZCNT)
      COMMON /B25/ALPHA(5)
      COMMON /IO1/IOUT1,INPUT1
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
      DIMENSION ALPHAO(6)
      INTEGER TIME1,TIME2,TIMDIF
      CALL RTIME(TIME2)
      TIMDIF=TIME2-TIME1
C     READ(INPUT1,140) (ALPHAO(I),I=1,6)
  140 FORMAT (15X,6A5)
C     WRITE(IOUT1,150)(ALPHAO(I),I=1,6),IZCNT,TIMDIF
  150 FORMAT(6A5,'|',1X,I7,1X,I7,2X)
      RETURN
      END
C*** MATFOM
      SUBROUTINE MATFOM(PSOL)
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
      COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
      COMMON /B11/BASIS(50),DBASIS(50)
      DIMENSION BASE(50,50),PSOL(50),DSOL(50)
      INTEGER BASIS,IPSOL(50),DBASIS
      M1=M+N-1;M2=M*N
      DO 10 I = 1,M2
   10 IPSOL(I)=PSOL(I)+.005
      J=0
      DO 20 I = 1,M2
```

```
      IF(IPSOL(I).LE.0.0)GO TO 20
      J=J+1
      BASIS(J)=I
   20 CONTINUE
      IF(J.EQ.M1)GO TO 40
      DO 35 I = 1,M2
      DO 25 J = 1,M1
      IF(BASIS(J).EQ.I)GO TO 35
   25 CONTINUE
      BASIS(M1)=I
      GO TO 40
   35 CONTINUE
   40 DO 45 J = 1,M1
      DBASIS(J)=BASIS(J)
   45 BASE(I,J)=A(I,BASIS(J))
      CALL INVERT(BASE,M1)
      CALL MATMLT(BASE,A,B,M1,M2)
      RETURN
      END
C
C
C
C
C
C***
C
      SUBROUTINE INVERT (A,N)
      INTEGER PIVR,PIVC
      DIMENSION A(50,50),PIVR(50),PIVC(50),PIVE(50),Y(50)
      DO 4 I=1,N
      PIVR(I)=0
    4 PIVC(I)=0
      K=1
    5 PIVE(K)=0.0
      DO 9 I=1,N
      DO 6 L=1,K
      IF (I.EQ.PIVR(L)) GO TO 9
    6 CONTINUE
      DO 8 J=1,N
      DO 7 L=1,K
      IF (J.EQ.PIVC(L)) GO TO 8
    7 CONTINUE
      AB=ABS(A(I,J))
      C=ABS(PIVE(K))
      IF (AB.LT.C) GO TO 8
      II=I
      JJ=J
      PIVE(K)=A(I,J)
    8 CONTINUE
    9 CONTINUE
      PIVR(K)=II
      PIVC(K)=JJ
      DO 11 J=1,N
   11 A(II,J)=A(II,J)/PIVE(K)
      A(II,JJ)=1./PIVE(K)
      DO 13 I=1,N
      B=A(I,JJ)
      IF (I.EQ.II) GO TO 13
      A(I,JJ)=-B/PIVE(K)
      DO 12 J=1,N
      IF (J.EQ.JJ) GO TO 12
      A(I,J)=A(I,J)-B*A(II,J)
   12 CONTINUE
   13 CONTINUE
      K=K+1
      IF (K.LE.N) GO TO 5
      DO 16 J=1,N
      DO 15 I=1,N
      IR=PIVR(I)
      IC=PIVC(I)
```

```
 15     Y(IC)=A(IR,J)
        DO 16I=1,N
 16     A(I,J)=Y(I)
        DO 20 I=1,N
        DO 19 J=1,N
        JR=PIVR(J)
        JC=PIVC(J)
 19     Y(JR)=A(I,JC)
        DO 20 J=1,N
 20     A(I,J)=Y(J)
        RETURN
        END
C
C***
C
        SUBROUTINE MATMLT(BASE,A,B,M1,M2)
        DIMENSION BASE(50,50),A(50,50),B(50),A1(50,50),B1(50)
        DO 30  I = 1,M1
        DO 25 J = 1,M2
        SUM = 0.0
        DO 20 K = 1,M1
  20    SUM=SUM+BASE(I,K)*A(K,J)
  25    A1(I,J)=SUM
  30    CONTINUE
        DO 45 I = 1,M1
        SUM=0.0
        DO 35 J = 1,M1
  35    SUM = SUM + BASE(I,J)*B(J)
  45    B1(I)=SUM
        DO 55 I = 1,M1
        B(I)=B1(I)
        DO 55 J = 1,M2
  55    A(I,J)=A1(I,J)
        RETURN
        END
C
        SUBROUTINE COMPUT
        COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
        COMMON /B11/BASIS(50),IBASIS(50)
        COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
        COMMON /B6/X(50,50)
        COMMON /B13/IZCNT,IZ(50,2)
        DIMENSION PSOL(50),COST(50)
        INTEGER BASIS,DBASIS
        M1=M+N-1;M2=M*N
        DO 5 I = 1,M2
        COST(I)=0.0
        PSOL(I)=0.0
   5    CONTINUE
        DO 15 J = 1,M1
  15    PSOL(BASIS(J))=B(J)
        CALL XCHANG(PSOL,COST,X,M2,SUM)
        IZCNT=IZCNT+1
        SUM1=0.0;SUM2=0.0
        DO 25 I = 1,M
        DO 25 J =1,N
        SUM1=SUM1+X(I,J)*ICOST1(I,J)
        SUM2=SUM2+X(I,J)*ICOST2(I,J)
  25    CONTINUE
        IZ(IZCNT,1)=SUM1+0.5
        IZ(IZCNT,2)=SUM2+0.5
        CALL PRINT(2);CALL PRINT(14)
        RETURN
        END
C
C*** FNDENT ***
C
        SUBROUTINE FNDENT(ENTER,STOP)
        COMMON /IO/INPUT,IOUT
        COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
```

```
      COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
      COMMON /B11/BASIS(50),IBASIS(50)
      LOGICAL STOP
      DIMENSION DJBAR(50),CJBAR(50)
      INTEGER ENTER,ROW,COL,BASIS,DBASIS
      STOP=.FALSE.
      M1=M+N-1;M2=M*N
      ENTER=0
      FMIN=9999.
      DO 5 I=1,M2
      CJBAR(I)=0.0
    5 DJBAR(I)=0.0
      DO 40 J=1,M2
      DO 15 I = 1,M1
      IF(J.EQ.BASIS(I))GO TO 40
   15 CONTINUE
      DUMMY1=0.0;DUMMY2=0.0
      DO 20 I =1,M1
      ROW=(BASIS(I)-1)/N+1
      COL=BASIS(I)-((ROW-1)*N)
      DUMMY1=DUMMY1+ICOST1(ROW,COL)*A(I,J)
      DUMMY2=DUMMY2+ICOST2(ROW,COL)*A(I,J)
   20 CONTINUE
      ROW=(J-1)/N+1
      COL=J-((ROW-1)*N)
      CJBAR(J)=ICOST1(ROW,COL)-DUMMY1
      DJBAR(J)=ICOST2(ROW,COL)-DUMMY2
      IF(CJBAR(J).GE.0.AND.DJBAR(J).LT.0.0) GO TO 22
      GO TO 40
   22 IF(FMIN-(-CJBAR(J)/DJBAR(J)))40,28,25
   25 ENTER=J
      FMIN=-CJBAR(J)/DJBAR(J)
   28 XMIN1=9999.0;XMIN2=9999.0
      DO 35 I = 1,M1
      IF(A(I,ENTER).LE.0.0)GO TO 30
      RATIO1=B(I)/A(I,ENTER)
      IF(RATIO1.GE.XMIN1)GO TO 30
      XMIN1=RATIO1
   30 IF(A(I,J).LE.0.0)GO TO 35
      RATIO1=B(I)/A(I,J)
      IF(RATIO1.GE.XMIN2)GO TO 35
      XMIN2=RATIO1
   35 CONTINUE
      IF(XMIN1.LT.9999.0.AND.XMIN2.LT.9999.0)GO TO 36
      GO TO 40
   36 MIN=ENTER
      IF(CJBAR(J)*XMIN2.GE.DJBAR(ENTER)*XMIN1)MIN=J
      ENTER= MIN
   40 CONTINUE
      WRITE(IOUT,104)(BASIS(I),I=1,M1)
  104 FORMAT('BASIS:',20(I5,1X))
      WRITE(IOUT,101)(CJBAR(J),J=1,M2)
      WRITE(IOUT,102)(DJBAR(J),J=1,M2)
C     DO 40 I = 1,M1
C  40 WRITE(IOUT,103)(A(I,J),J=1,M2),B(I)
  101 FORMAT(' CJBAR:',20(F5.1,1X))
  102 FORMAT(' DJBAR:',20(F5.1,1X))
  103 FORMAT(' A MAT:',20(F5.1,1X))
      IF(ENTER.NE.0)RETURN
      STOP=.TRUE.
      RETURN
      END
C
C*** CDLEAV ***
C
      SUBROUTINE FDLEAV(ENTER,LEAVE,UBCOND)
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5(
      COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
      COMMON /B11/BASIS(50),DBASIS(50)
      INTEGER BASIS,DBASIS,ENTER,LEAVE
```

```
      LOGICAL UBCOND
      XMIN=9999.0;UBCOND=.TRUE.
      M1=M+N-1;M2=M*N
      DO 10 I = 1,M1
      IF(A(I,ENTER).LE.0.0)GO TO 10
      RATIO=B(I)/A(I,ENTER)
      IF(RATIO.GE.XMIN)GO TO 10
      XMIN=RATIO
      UBCOND=.FALSE.
      LEAVE=I
 10   CONTINUE
      RETURN
      END
C
C***
C
      SUBROUTINE ERROR(I)
      COMMON /IO/INPUT,IOUT
      GO TO (10,15,10) I
 10   WRITE(IOUT,101)
 101  FORMAT(15X,'INFEASIBLE SOLUTION OR WRONG DATA'//)
      GO TO 20
 15   WRITE(IOUT,102)
 102  FORMAT(15X,'UNBOUNDED SOLUTION'//)
 20   X=-1
      ROOT=SQRT(X)
      END
C
C*** SUBROUTINE DREAD ***
C
      SUBROUTINE DREAD(END)
      LOGICAL END
      COMMON /IO/INPUT,IOUT
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
      COMMON /B25/ALPHA(5)
      END = .FALSE.
      READ(INPUT,112,END=100)(ALPHA(I),I=1,5), M,INTO,N
 112  FORMAT(5A5,I4,A2,I4)
      READ(INPUT,101) (MA(I),I=1,M)
      READ(INPUT,101) (MB(J),J=1,N)
      DO 155 J = 1,M
 155  READ(INPUT,101)(ICOST1(J,K),K=1,N)
      DO 156 J = 1,M
 156  READ(INPUT,101)(ICOST2(J,K),K=1,N)
 101  FORMAT(40I3)
      WRITE(IOUT,113)(ALPHA(I),I=1,5),M,INTO,N
 113  FORMAT(/////5A5,I4,A2,I4///)
      DO 15 II=1,2
      WRITE(IOUT,102) II,(I,I=1,N)
      IF(II.EQ.1)WRITE(IOUT,103) (ICOST1(1,J),J=1,N)
      IF(II.EQ.2)WRITE(IOUT,103) (ICOST2(1,J),J=1,N)
 102  FORMAT(1H1,30X,'THE INPUT COST MATRIX ',I5,//10X,'DESTN
     3,5X,20I5)
 103  FORMAT(5X,'SOURCE  1'9X,20I5)
      DO 10 I = 2,M
      IF(II.EQ.2)WRITE(IOUT,104)I,(ICOST2(I,J),J=1,N)
      IF(II.EQ.1)WRITE(IOUT,104)I,(ICOST1(I,J),J=1,N)
 104  FORMAT (10X,I4,9X,20I5/)
 10   CONTINUE
 15   CONTINUE
      WRITE(IOUT,106) (MA(I),I=1,M)
 106  FORMAT(10X,'SUPPLY' 7X,20I5)
 107  FORMAT(10X,'DEMAND',7X,20I5)
      WRITE(IOUT,107)(MB(J),J=1,N)
      WRITE(IOUT,108)
 108  FORMAT(1H1,30X,'Solutions by Aneja and Nair method'//)
      RETURN
 100  END=.TRUE.
      RETURN
      END
```

```
C
C***
C
      SUBROUTINE SETUP
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
      COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
      M1=M+N-1
      M2=M*N
      DO 10 I = 1,M
      B(I)=MA(I)
      DO 10 J = 1,N
      IJ= (I-1)*N + J
   10 A(I,IJ)=1.0
      DO 15 I = M+1,M1
      B(I)=MB(I-M)
      DO 15 J = 1,M
      IJ=(J-1)*N + (I-M)
   15 A(I,IJ)=1.0
      RETURN
      END
C
C***
C
      SUBROUTINE PRINT(CODE)
      COMMON /B6/X(50,50)
      COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
      COMMON /IO/INPUT,IOUT
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
C     COMMON /B13/IZCNT,IZ(50,2)
      COMMON /B4/A(50,150),B(50),IX,RW(4000),IW(2000),LPCNT
      COMMON /B14/MINCOC(2),MINCOD(2)
      INTEGER CODE
      REAL MINCOC,MINCOD
      GO TO(10,20,30,40,50,60,70,800,900,1000,1000,1000,1000,
   10 WRITE(IOUT,110),IZ1(1),IZ2(1),IZ1(2),IZ2(2)
  110 FORMAT(15X,'Superior solution exists'/20X,'Z(1,1)='I5,'
     1,I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
      RETURN
   20 WRITE(IOUT,120) (X(1,J),J=1,N)
      DO 25 I = 2,M
      WRITE(IOUT,125) (X(I,J),J=1,N)
   25 CONTINUE
  120 FORMAT(5X,'The solution matrix follows'//5X,20(F6.1,2X)
  125 FORMAT(5X,20(F6.1,2X))
      RETURN
   30 WRITE(IOUT,130)(X(1,J),J=1,N)
      DO 35 I = 2,M
      WRITE(IOUT,135) (X(I,J),J=1,N)
   35 CONTINUE
  130 FORMAT(5x,'The shipping matrix follows'/5X,20(F7.1,2X).
  135 FORMAT(5X,20(F7.1,2X))
      RETURN
   40 WRITE(IOUT,140)
  140 FORMAT(10X,'This is not an efficient point'//)
      RETURN
   50 WRITE(IOUT,150)
  150 FORMAT(50X,'THE PROBLEM IS TERMINATED '//)
      WRITE(IOUT,245) LPCNT
  245 FORMAT(20X,'No of LPs solved:',I6/20X,'~~ ~~ ~~~ ~~~~~~
      WRITE(IOUT,250) IZCNT
  250 FORMAT(40X,'No. of efficient points are:',I5,/40X,29('_
      WRITE(IOUT,255)
  255 FORMAT(30X,'Points follow:'/30x,'------ ------'//30x,'
     1wo'//)
      WRITE(IOUT,256)(IZ1(I),IZ2(I),I=1,IZCNT)
  256 FORMAT((29X,I5,5X,I5/))
      RETURN
   60 WRITE(IOUT,160)IZCNT,IZ1(IZCNT),IZCNT,IZ2(IZCNT)
  160 FORMAT(20X,'The efficient points are Z(',I3,',1)= ',I5
     15X,'Z(',I3,',2)= ',I5)
```

```
      RETURN
   70 WRITE(IOUT,170)IZ1(1),IZ2(1),IZ1(2),IZ2(2)
  170 FORMAT(15X,'Initial extreme points follow'/20X,'Z(1,1)=
     1' Z(1,2)=',I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
      RETURN
  800 WRITE(IOUT,805)
  805 FORMAT(50X,'Solutions by parametrically varying r.h.sid
      WRITE(IOUT,120)(X(1,J),J=1,N)
      DO 810 I = 2,M
      WRITE(IOUT,125)(X(I,J),J=1,N)
  810 CONTINUE
      IF(IZCNT==2)GO TO 820
      WRITE(IOUT,815) IZ1(IZCNT),IZ2(IZCNT)
  815 FORMAT(20X,'MIN C(I,J)*X(I,J)',F10.4,'CORRESPNG D(I,J)*
     1,5X,F10.4)
      RETURN
  820 WRITE(IOUT,10005)
10005 FORMAT(50X,'The primal is infeasbible'//50x,'The proble
     2ted and next problem pursued')
      RETURN
 1000 WRITE(IOUT,1105)
      RETURN
  900 WRITE(IOUT,1105)
 1105 FORMAT(50X,' WHOLE RANGE COVERED')
      RETURN
      END
C
C***
C
```

APPENDIX B

Program Listing for Aneja and Nair's Algorithm

```
C
C  PROGRAM MAIN ROUTINE
C
C  **** VARABLE DECLARATION
         COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
         COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
         COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
         COMMON /IO/INPUT,IOUT
         COMMON /B25/ALPHA(5)
         COMMON /I01/IOUT1
         INTEGER TIME1,TIME2,TIMDIF
         LOGICAL END,ILEMTY,INFCON,SUPRIR,STOP
         DATA INPUT,IOUT,IOUT1/20,21,22/
         IA=50
         OPEN(UNIT=INPUT,DEVICE=DSK,FILE='TRANSP.DAT')
         OPEN(UNIT=IOUT,FILE='TRANSP.OUT')
C        OPEN(UNIT=IOUT,FILE='TRANSP.OUT',ACCESS='APPEND')
         OPEN(UNIT=IOUT1,FILE='TIMING.REP',ACCESS='APPEND')
C
  1      CALL RTIME(TIME1)
         CALL ZERO
         CALL DREAD (END)
         IF(END) GO TO 1500
         CALL FEACHK(INFCON)
         IF(INFCON) CALL ERROR(1)
C
C***   SET UP A MATRIX WITH EQUALITY CONSTRAINTS
C
         CALL SETUP
C
C*** FIND INITIAL EXTREME POINTS
C
         CALL INITAL(SUPRIR)
         IF(SUPRIR) 11,10
  11     CALL TIME(TIME1,1)
         CALL PRINT(1)
         GO TO 1
C
C***    CHOOSE NEXT IL ELEMENT. IF IL IS EMPTY TAKE NEXT PROBLEM
C
  10     CALL PRINT(7)
  100    CALL CHOSIL(ILEMTY,IPOSTN)
         IF(ILEMTY) GO TO 1000
C
C*** FIND THE SOLUTION TO THE TRANSPORTATION PROBLEM WITH THIS
C*** IL ELEMENT. IF ALTERNATIVE OPTIMA EXITS CHOOSE MIN SIGMA C * X
C
         CALL STEP21 (IPOSTN,SUM1,SUM2,IR,IS,STOP)
         IF(STOP)GO TO 1
C
C*** DO THE SECOND PART OF THE SETP 2
C*** IF THE SOLUTION IS NOT EFFEICIENT DO NO RECORD
C*** ELSE RECORD. DO THIS  BY STEP22
C
         CALL SETP22(SUM1,SUM2,IR,IS)
         CALL SETP3(IR,IS)
         GO TO 100
 1000    CONTINUE
         IFIVE=5
         CALL PRINT(IFIVE)
         CALL TIME(TIME1,IZCNT)
         GO TO 1
 1500    STOP
         CLOSE(UNIT=INPUT)
```

```
      CLOSE(UNIT=IOUT)
      CALL UERTST
      CALL ZX1LP
      END




      SUBROUTINE ZERO
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
      COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
      COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
      COMMON /B11/BASIS(50),IBASIS(50)
      COMMON /B13/IZCNTD,IZ(50,2)
      COMMON /B6/X(50,50)
      DO 10 I = 1,50
      MA(I)=0;MB(I)=0
      B(I)=0
      BASIS(I)=0
      IBASIS(I)=0
      IZ1(I)=0
      IZ2(I)=0
      IZ(I,1)=0;IZ(I,2)=0
      DO 10 J = 1,50
      A(I,J)=0.0
      X(I,J)=0.0
 10   CONTINUE
      M=0;N=0;IZCNT=0
      ILCNT=0
      IZCNTD=0
      LPCNT=0
      DO 15 I = 1,4000
      RW(I)=0.0
 15   CONTINUE
      DO 20 J = 1,2000
 20   IW(J)=0
      RETURN
      END
C


      SUBROUTINE TIME(TIME1,IZCNT)
      COMMON /B25/ALPHA(5)
      COMMON /IO1/IOUT1
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
      INTEGER TIME1,TIME2,TIMDIF
      CALL RTIME(TIME2)
      TIMDIF=TIME2-TIME1
      WRITE(IOUT1,150)(ALPHA(I),I=1,5),M,N,TIMDIF,IZCNT
 150  FORMAT(5A5,2X,I2,'x',I2,2X,I7,1X,I3,20X)
      RETURN
      END
C
C*** SUBROUTINE DREAD ***
C



      SUBROUTINE DREAD(END)
      LOGICAL END
      COMMON /IO/INPUT,IOUT
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
      COMMON /B25/ALPHA(5)
      END = .FALSE.
      READ(INPUT,112,END=100)(ALPHA(I),I=1,5), M,INTO,N
 112  FORMAT(5A5,I4,A2,I4)
      READ(INPUT,101) (MA(I),I=1,M)
      READ(INPUT,101) (MB(J),J=1,N)
      DO 155 J = 1,M
```

```
  155 READ(INPUT,101)(ICOST1(J,K),K=1,N)
      DO 156 J = 1,M
  156 READ(INPUT,101)(ICOST2(J,K),K=1,N)
  101 FORMAT(40I3)
      WRITE(IOUT,113)(ALPHA(I),I=1,5),M,INTO,N
  113 FORMAT(/////5A5,I4,A2,I4///)
      DO 15 II=1,2
      WRITE(IOUT,102) II,(I,I=1,N)
      IF(II.EQ.1)WRITE(IOUT,103) (ICOST1(1,J),J=1,N)
      IF(II.EQ.2)WRITE(IOUT,103) (ICOST2(1,J),J=1,N)
  102 FORMAT(1H1,30X,'THE INPUT COST MATRIX ',I5,//10X,'DESTN.->'
     3,5X,20I5)
  103 FORMAT(5X,'SOURCE  1'9X,20I5)
      DO 10 I = 2,M
      IF(II.EQ.2)WRITE(IOUT,104)I,(ICOST2(I,J),J=1,N)
      IF(II.EQ.1)WRITE(IOUT,104)I,(ICOST1(I,J),J=1,N)
  104 FORMAT (10X,I4,9X,20I5/)
  10  CONTINUE
   15 CONTINUE
      WRITE(IOUT,106) (MA(I),I=1,M)
  106 FORMAT(10X,'SUPPLY' 7X,20I5)
  107 FORMAT(10X,'DEMAND',7X,20I5)
      WRITE(IOUT,107)(MB(J),J=1,N)
      WRITE(IOUT,108)
  108 FORMAT(1H1,30X,'Solutions by Aneja and Nair method'//)
      RETURN
  100 END=.TRUE.
      RETURN
      END
C
C***
C



      SUBROUTINE SETUP
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
      COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
      M1=M+N-1
      M2=M*N
      DO 10 I = 1,M
      B(I)=MA(I)
      DO 10 J = 1,N
      IJ= (I-1)*N + J
   10 A(I,IJ)=1.0
      DO 15 I = M+1,M1
      B(I)=MB(I-M)
      DO 15 J = 1,M
      IJ=(J-1)*N + (I-M)
   15 A(I,IJ)=1.0
      RETURN
      END
C
C***
C
C
C***
C



      SUBROUTINE INITAL(SUPRIR)
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
      COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
      COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
      COMMON /B6/X(50,50)
      DIMENSION COST(150),PSOL(150),DSOL(150)
      LOGICAL SUPRIR
      INTEGER TWO
      M1=M+N-1
```

```
      M2=M*N
      DO 100 II=1,2
      DO 15 I =1,M
      DO 15 J =1,N
      IJ=(I-1)*N+J
      IF(II.EQ.2)COST(IJ)=-ICOST2(I,J)
      IF(II.EQ.1)COST(IJ)=-ICOST1(I,J)
15    CONTINUE
      CALL ZX3LP(A,IA,B,COST,M2,0,M1,S,PSOL,DSOL,RW,IW,IER)
      LPCNT=LPCNT+1
      CALL XCHANG(PSOL,COST,X,M2,TOTCOS)
      TWO=2
      CALL PRINT(TWO)
      IF(II.EQ.1)IZ1(II)=-TOTCOS
      IF(II.EQ.2)IZ2(II)=-TOTCOS
      B(M1+1)=-TOTCOS
      DO 25 I = 1,M
      DO 25 J = 1,N
      IJ=(I-1)*N +J
      A(M1+1,IJ)=ICOST2(I,J)
      IF(II.EQ.1)A(M1+1,IJ)=ICOST1(I,J)
      COST(IJ)=-ICOST2(I,J)
      IF(II.EQ.2) COST(IJ)=-ICOST1(I,J)
25    CONTINUE
      LPCNT=LPCNT+1
      CALL ZX3LP(A,IA,B,COST,M2,0,M1+1,S,PSOL,DSOL,RW,IW,IER)
      CALL XCHANG(PSOL,COST,X,M2,TOTCOS)
28    CONTINUE
      IF(II.EQ.1)IZ2(II)=-TOTCOS
      IF(II.EQ.2)IZ1(II)=-TOTCOS
      TWO = 2
      CALL PRINT(TWO)
100   CONTINUE
      ILCNT=1
      IL(1,1)=1
      IL(1,2)=2
      IZCNT=2
      SUPRIR=.FALSE.
      IF(IZ1(1).EQ.IZ1(2).AND.IZ2(1).EQ.IZ2(2)) SUPRIR=.TRUE.
      RETURN
      END
C
C***
C



      SUBROUTINE STEP21(IPOSTN,SUM1,SUM2,IR,IS,STOP)
      COMMON /IO/INPUT,IOUT
      COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
      COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
      COMMON /B6/X(50,50)
      COMMON /B25/ALPHA(5)
      DIMENSION COST(150),PSOL(150),DSOL(150)
      LOGICAL STOP
      INTEGER THREE ,TWO
      STOP = .FALSE.
      THREE = 3
      TWO=2
      IR=IL(IPOSTN,1)
      IS=IL(IPOSTN,2)
      IA1=ABS(IZ2(IS)-IZ2(IR))
      IA2=ABS(IZ1(IS)-IZ1(IR))
      DO 10 I =1,M
      DO 10 J = 1,N
      IJ=(I-1)*N+J
      COST(IJ)=IA1*ICOST1(I,J) + IA2*ICOST2(I,J)
10    X(I,J)=COST(IJ)
      DO 15 I = 1,M*N
```

```
   15     COST(I)=-COST(I)
          CALL PRINT(THREE)
          M1=M+N-1
          M2=M*N
          LPCNT=LPCNT+1
          CALL ZX3LP(A,IA,B,COST,M2,0,M1,TOTCOS,PSOL,DSOL,RW,IW,IER)
   25     DO 35 I = 1,M
          DO 35 J = 1,N
          IJ=(I-1)*N+J
          A(M+N,IJ)=IA1*ICOST1(I,J) + IA2*ICOST2(I,J)
   35     COST (IJ)=-ICOST1(I,J)
          B(M+N)=-TOTCOS
          LPCNT=LPCNT+1
          CALL ZX3LP(A,IA,B,COST,M2,0,M1+1,S,PSOL,DSOL,RW,IW,IER)
          IF(IER.EQ.133)GO TO 55
   30     SUM1=0.0;SUM2=0.0
          CALL XCHANG(PSOL,COST,X,M2,TOTCOS)
C         CALL PRINT(TWO)
          DO 50 I =1,M
          DO 50 J = 1,N
          SUM1=SUM1+ICOST1(I,J)*X(I,J)
   50     SUM2=SUM2+ICOST2(I,J)*X(I,J)
          RETURN
   55     STOP=.TRUE.
          TYPE 601,(ALPHA(III),III=1,5)
  601     FORMAT(//5A5,' IS INFEASIBLE'// )
          CALL PRINT(14)
          RETURN
          END
C***
C



          SUBROUTINE SETP22 (SUM1,SUM2,IR,IS)
          COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
          COMMON /B6/X(50,50)
          REAL SUM1,SUM2
          INTEGER FOUR,SIX
          FOUR=4;SIX=6
          ISUM1=SUM1;ISUM2=SUM2
          DIFR1=ABS(SUM1-IZ1(IR))
          DIFR2=ABS(SUM2-IZ2(IR))
          DIFS1=ABS(SUM1-IZ1(IS))
          DIFS2=ABS(SUM2-IZ2(IS))
          IF((DIFR1.LT.1.AND.DIFR2.LT.1).OR.(DIFS1.LT.1.AND.DIFS2.LT.1
          1))25,20
C   25    CALL PRINT(FOUR)
C*** WRITES THAT THIS POINT IS NOT EFFICIENT POINT
   25     RETURN
   20     IZCNT=IZCNT+1
          IZ1(IZCNT)=ISUM1
          IZ2(IZCNT)=ISUM2
          ILCNT=ILCNT+1
          IL(ILCNT,1)=IR
          IL(ILCNT,2)=IZCNT
          ILCNT=ILCNT+1
          IL(ILCNT,1)=IZCNT
          IL(ILCNT,2)=IS
C***     RECORDS THE EFFICIENT POINT
          CALL PRINT(2)
          CALL PRINT(SIX)
          RETURN
          END
C
C***
C
```

```
      SUBROUTINE SETP3 (IR,IS)
      COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
      DO 10 I =1,ILCNT
      IF((IL(I,1).EQ.IR).AND.(IL(I,2).EQ.IS))GO TO 20
   10 CONTINUE
      RETURN
   20 IL(I,1)=0
      IL(I,2)=0
      RETURN
      END
C
C***
C



      SUBROUTINE ERROR(I)
      COMMON /IO/INPUT,IOUT
      GO TO (10,15,10) I
   10 WRITE(IOUT,101)
  101 FORMAT(15X,'INFEASIBLE SOLUTION OR WRONG DATA'//)
      RETURN
   15 WRITE(IOUT,102)
  102 FORMAT(15X,'UNBOUNDED SOLUTION'//)
      RETURN
      END
C
C***
C



      SUBROUTINE PRINT(CODE)
      COMMON /B6/X(50,50)
      COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
      COMMON /IO/INPUT,IOUT
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
C     COMMON /B13/IZCNT,IZ(50,2)
      COMMON /B4/A(50,150),B(50),IX,RW(4000),IW(2000),LPCNT
      COMMON /B14/MINCOC(2),MINCOD(2)
      INTEGER CODE
      REAL MINCOC,MINCOD
      GO TO(10,20,30,40,50,60,70,800,900,1000,1000,1000,1000,820)(
   10 WRITE(IOUT,110),IZ1(1),IZ2(1),IZ1(2),IZ2(2)
  110 FORMAT(15X,'Superior solution exists'/20X,'Z(1,1)='I5,' Z(1,
     1,I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
      RETURN
   20 WRITE(IOUT,120) (X(1,J),J=1,N)
      DO 25 I = 2,M
      WRITE(IOUT,125) (X(I,J),J=1,N)
   25 CONTINUE
  120 FORMAT(5X,'The solution matrix follows'//5X,20(F6.1,2X)/)
  125 FORMAT(5X,20(F6.1,2X))
      RETURN
   30 WRITE(IOUT,130)(X(1,J),J=1,N)
      DO 35 I = 2,M
      WRITE(IOUT,135) (X(I,J),J=1,N)
   35 CONTINUE
  130 FORMAT(5x,'The shipping matrix follows'/5X,20(F7.1,2X)//)
  135 FORMAT(5X,20(F7.1,2X))
      RETURN
   40 WRITE(IOUT,140)
  140 FORMAT(10X,'This is not an efficient point'//)
      RETURN
   50 WRITE(IOUT,150)
  150 FORMAT(50X,'THE PROBLEM IS TERMINATED '//)
      WRITE(IOUT,245) LPCNT
  245 FORMAT(20X,'No of LPs solved:',I6/20X,'~~ ~~ ~~~ ~~~~~~~'//)
      WRITE(IOUT,250) IZCNT
  250 FORMAT(40X,'No. of efficient points are:',I5,/40X,29('_')//
```

```
      WRITE(IOUT,255)
255   FORMAT(30X,'Points follow:'/30x,'------ ------'//30x,'One',8x,'T
     1wo'//)
      WRITE(IOUT,256)(IZ1(I),IZ2(I),I=1,IZCNT)
256   FORMAT((29X,I5,5X,I5/))
      RETURN
60    WRITE(IOUT,160)IZCNT,IZ1(IZCNT),IZCNT,IZ2(IZCNT)
160   FORMAT(20X,'The efficient points are Z(',I3,',1)= ',I5,
     15X,'Z(',I3,',2)= ',I5)
      RETURN
70    WRITE(IOUT,170)IZ1(1),IZ2(1),IZ1(2),IZ2(2)
170   FORMAT(15X,'Initial extreme points follow'/20X,'Z(1,1)='I5,
     1' Z(1,2)=',I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
      RETURN
800   WRITE(IOUT,805)
805   FORMAT(50X,'Solutions by parametrically varying r.h.side'//)
      WRITE(IOUT,120)(X(1,J),J=1,N)
      DO 810 I = 2,M
      WRITE(IOUT,125)(X(I,J),J=1,N)
810   CONTINUE
      IF(IZCNT==2)GO TO 820
      WRITE(IOUT,815) IZ1(IZCNT),IZ2(IZCNT)
815   FORMAT(20X,'MIN C(I,J)*X(I,J)',F10.4,'CORRESPNG D(I,J)*X(I,J)'
     1,5X,F10.4)
      RETURN
820   WRITE(IOUT,10005)
10005 FORMAT(50X,'The primal is infeasbible'//50x,'The problem is abor
     2ted and next problem pursued')
      RETURN
1000  WRITE(IOUT,1105)
      RETURN
900   WRITE(IOUT,1105)
1105  FORMAT(50X,' WHOLE RANGE COVERED')
      RETURN
      END
C
C***
C



      SUBROUTINE FEACHK(INFCON)
      LOGICAL INFCON
      COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
      MSHRT=0;MSUP=0
      INFCON=.TRUE.
      DO 10 I = 1,M
      MSUP=MSUP + MA(I)
      MSHRT=0
      DO 10 J=1,N
      IF((MA(I).LT.0).OR.(MB(J).LT.0).OR.(ICOST1(I,J).LT.0).OR.
     1(ICOST2(I,J).LT.0))GO TO 15
10    MSHRT = MSHRT + MB(J)
      IF(MSHRT.NE.MSUP)  GO TO 15
      INFCON=.FALSE.
15    RETURN
      END
C
C***
C


      SUBROUTINE CHOSIL (ILEMTY,IPOSTN)
      COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
      COMMON /IO/INPUT,IOUT
      INTEGER IPOSTN
      LOGICAL ILEMTY
      ILEMTY=.FALSE.
C     WRITE(IOUT,505),ILCNT
505   FORMAT(' ILCNT=',I5)
```

```
      DO 100 I = 1,ILCNT
      IF(IL(I,1).GT.0) GO TO 110
100   CONTINUE
      ILEMTY=.TRUE.
      RETURN
110   IPOSTN=I
      RETURN
      END
C
C***
C



      SUBROUTINE XCHANG(PSOL,COST,X,N,SUM)
      COMMON /B1/NOSP,NOSHRT,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
      DIMENSION PSOL(N),COST(N),X(50,50)
      INTEGER ROW,COL
      SUM=0.0
      DO 30 I = 1,N
      IP=PSOL(I)+0.5
      PSOL(I)=IP
      ROW=(I-1)/NOSHRT +1
      COL=I-((ROW-1)*NOSHRT)
      SUM=SUM + PSOL(I)*COST(I)
      X(ROW,COL)=PSOL(I)
30    CONTINUE
      RETURN
      END
```

## APPENDIX C

### Program for Generation of Random Problems

```
c With ISEED 2948513, ten problems of 4 by 4 were generated.
c witn ISEED 4548511, twenty problems of 5 by 5 were generated.
C with ISEED 2948513, twenty problems of 3 by 3 were generated.
        IMPLICIT INTEGER (A-Z)
        DIMENSION SUPPLY(50),DEMAND(50),COST1(50,50),COST2(50,50)
        REAL R(5000)
        NOOFPR=10
        IOUT=21
        OPEN(UNIT=IOUT,FILE='GENERT.OUT')
C   ISEED given for the first set of problems of 4 by 4 is 2948513
C   ISEED given for the second set of problems of 5 by 5 is 4548511
C   ISEED GIVEN FOR THE FOURTH SET OF PROBLEMS OF 7 BY 7 IS 40205096
C  ISEED GIVEN FOR THE FIFTH SET OF PROBLEMS OF 6 BY 6 IS 549504176
        ISEED=549504176
        NOTIME=1
        DO 95 I = 1,NOTIME
        IF(I.EQ.1)SIZE = 6
        IF(I.NE.1)SIZE=SIZE*2
        N=2*(SIZE**2)+2*SIZE
        DO 90 JJJ=1,NOOFPR
        DO 10 II = 1,N
   10   R(II)=0.0
        ISEED=ISEED+1
        CALL GGUB(ISEED,N,R)
        TOTAL=0; RINDEX=0
        DO 15 II = 1,SIZE
   12   RINDEX=RINDEX+1
        INTER=R(RINDEX)*100.0
        IF(INTER.EQ.0.OR.INTER.GT.15)GO TO 12
        SUPPLY(II)=INTER
        TOTAL=TOTAL+SUPPLY(II)
   15   CONTINUE
c       PAUSE ' FINISHED SUPPLY'
        DO 20 II=1,SIZE
        IF(II==SIZE)GO TO 19
   16   RINDEX=RINDEX+1
        IF(RINDEX==N)200,205
        ISEED=ISEED+2
  200   CALL GGUB(ISEED,N,R)
        RINDEX=0
        GO TO 16
  205   INTER=R(RINDEX)*100.0
        IF((INTER.EQ.0).OR.(INTER.GE.TOTAL).OR.(INTER.GT.15))16,18
        TYPE 1949,II,DEMAND(II)
 1949   FORMAT(' DEMAND(',I3,')',I4)
   18   DEMAND(II)=INTER
        TOTAL=TOTAL-INTER
        IF(TOTAL .EQ.1)CALL MODIFY(DEMAND,II,TOTAL)
        GO TO 20
   19   DEMAND(II)=TOTAL
   20   CONTINUE
c       PAUSE ' Finished demand'
        DO 50 II=1,SIZE
        DO 50 JJ=1,SIZE
        DO 45 J = 1,2,1
   31   RINDEX=RINDEX+1
        IF(RINDEX==N)35,40
        ISEED=ISEED+2
   35   CALL GGUB(ISEED,N,R)
        RINDEX=0
        GO TO 31
   40   INTER=R(RINDEX)*10.0
        IF(INTER.EQ.0)GO TO 31
        IF(J.EQ.1) COST1(II,JJ)=INTER
        IF(J.EQ.2) COST2(II,JJ)=INTER
```

```
   45   CONTINUE
   50   CONTINUE
        WRITE(IOUT,110) JJJ,SIZE,SIZE
  110   FORMAT(' Problem no. ',I4,' of size ',I3,' x ',I3)
  101   FORMAT(40I3)
        WRITE(IOUT,101)(SUPPLY(II),II=1,SIZE)
        WRITE(IOUT,101)(DEMAND(II),II=1,SIZE)
        DO 82 JN=1,2
        DO 82 JP=1,SIZE
        IF(JN==1)WRITE(IOUT,101)(COST1(JP,II),II=1,SIZE)
        IF(JN==2)WRITE(IOUT,101)(COST2(JP,II),II=1,SIZE)
   82   CONTINUE
c       PAUSE ' One problem is over'
   90   CONTINUE
   95   CONTINUE
        STOP
        END
        SUBROUTINE MODIFY(DEMAND,II,TOTAL)
        IMPLICIT INTEGER (A-Z)
        DIMENSION DEMAND(50)
        MAX=DEMAND(1)
        DO 10 I=1,II
        IF(DEMAND(I).GE.MAX)  GO TO 5
        GO TO 10
  5     MAX=DEMAND(I)
        INDEX = I
  10    CONTINUE
        DEMAND(INDEX)=1
        TEMP = MAX-DEMAND(INDEX)
        TOTAL=TOTAL + TEMP
        RETURN
        END
```

A63022

**Date Slip**

This book is to be returned on the date last stamped.

| | |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

CD 6.72.9